CRANGE

CROSS-REFERENCING SOURCE CODE WITH CLANG

Anurag Patel — Red Hat — github.com/crange/crange

TABLE OF CONTENTS

SOURCE CODE

```c
#include <stdio.h>
int Fibonacci(int);
int Fibonacci(int n) {
  if (n == 0)
    return 0;
  else if (n == 1)
    return 1;
  else
    return (Fibonacci(n-1) + Fibonacci(n-2));
}

int main() {
  int n = 12, i = 0, c;
  for ( c = 1 ; c <= n ; c++ ) {
    printf("%d\n", Fibonacci(i)); i++;
  }
  return 0;
}
```

Fibonacci series

TOOLS CONSIDERED

CTAGS

+ Fast, comprehensive language support

- Definitions only

DOXYGEN

+ X-Ref, XML output

- Slow

GNU GLOBAL

+ Fast, definitions, X-Ref

ANTLR

+ Seems powerful, LL(*) parser

- Java, grammar files, complicated

FLEX+BISON

\+ *Is* powerful

\- CS 143 - Compilers

CLANG

+ Declarations, definitions, X-Ref, range, statements, operators, expressions, literals

CLANG

CLANG

Compiler front end and static analyzer for C, C++, Objective-C.
Converts programs to AST and allows AST manipulation.

AST

```
[snip]
|-FunctionDecl 0x610a090 <fibonacci.c:2:1, col:18> Fibonacci 'int (int)'
| `-ParmVarDecl 0x6109fd0 <col:15> 'int'
|-FunctionDecl 0x610a1d0 prev 0x610a090 <line:3:1, line:10:1> Fibonacci 'int
| |-ParmVarDecl 0x610a150 <line:3:15, col:19> n 'int'
| `-CompoundStmt 0x6153a70 <col:22, line:10:1>
|   `-IfStmt 0x6153a40 <line:4:3, line:9:44>
|     |-<<<NULL>>>
|     |-BinaryOperator 0x610a2d8 <line:4:7, col:12> 'int' '=='
|     | |-ImplicitCastExpr 0x610a2c0 <col:7> 'int' <LValueToRValue>
|     | | `-DeclRefExpr 0x610a278 <col:7> 'int' lvalue ParmVar 0x610a150 'n'
|     | `-IntegerLiteral 0x610a2a0 <col:12> 'int' 0
|     |-ReturnStmt 0x610a320 <line:5:5, col:12>
|     | `-IntegerLiteral 0x610a300 <col:12> 'int' 0
|     `-IfStmt 0x6153a10 <line:6:8, line:9:44>
[snip]
```

$ clang -cc1 -ast-dump fibonacci.c

LIBCLANG

LIBCLANG

- Library for processing source code.
- Translate source to AST
- Resolve identifiers and symbols
- Expand macros
- Syntax highlighting
- Code completion

INDEXING AND X-REF (0/6)

```
#include <stdio.h>

int Fibonacci(int);
int Fibonacci(int n) {
  if (n == 0)
    return 0;
  else if (n == 1)
    return 1;
  else
    return (Fibonacci(n-1) + Fibonacci(n-2));
}

int main() {
  int n = 12, i = 0, c;
  for ( c = 1 ; c <= n ; c++ ) {
    printf("%d\n", Fibonacci(i)); i++;
  }
  return 0;
}
```

Walk the AST

INDEXING AND X-REF (1/6)

```
#include <stdio.h>

int Fibonacci(int);
int Fibonacci(int n) {
  if (n == 0)
    return 0;
  else if (n == 1)
    return 1;
  else
    return (Fibonacci(n-1) + Fibonacci(n-2));
}

int main() {
  int n = 12, i = 0, c;
  for ( c = 1 ; c <= n ; c++ ) {
    printf("%d\n", Fibonacci(i)); i++;
  }
  return 0;
}
```

Macros

INDEXING AND X-REF (2/6)

```
#include <stdio.h>

int Fibonacci(int);
int Fibonacci(int n) {
  if (n == 0)
    return 0;
  else if (n == 1)
    return 1;
  else
    return (Fibonacci(n-1) + Fibonacci(n-2));
}

int main() {
  int n = 12, i = 0, c;
  for ( c = 1 ; c <= n ; c++ ) {
    printf("%d\n", Fibonacci(i)); i++;
  }
  return 0;
}
```

Declarations

INDEXING AND X-REF (3/6)

```
#include <stdio.h>

int Fibonacci(int);
int Fibonacci(int n) {
  if (n == 0)
    return 0;
  else if (n == 1)
    return 1;
  else
    return (Fibonacci(n-1) + Fibonacci(n-2));
}

int main() {
  int n = 12, i = 0, c;
  for ( c = 1 ; c <= n ; c++ ) {
    printf("%d\n", Fibonacci(i)); i++;
  }
  return 0;
}
```

References

INDEXING AND X-REF (4/6)

```
#include <stdio.h>

int Fibonacci(int);
int Fibonacci(int n) {
  if (n == 0)
    return 0;
  else if (n == 1)
    return 1;
  else
    return (Fibonacci(n-1) + Fibonacci(n-2));
}

int main() {
  int n = 12, i = 0, c;
  for ( c = 1 ; c <= n ; c++ ) {
    printf("%d\n", Fibonacci(i)); i++;
  }
  return 0;
}
```

Statements

INDEXING AND X-REF (5/6)

```
#include <stdio.h>

int Fibonacci(int);
int Fibonacci(int n) {
  if (n == 0)
    return 0;
  else if (n == 1)
    return 1;
  else
    return (Fibonacci(n-1) + Fibonacci(n-2));
}

int main() {
  int n = 12, i = 0, c;
  for ( c = 1 ; c <= n ; c++ ) {
    printf("%d\n", Fibonacci(i)); i++;
  }
  return 0;
}
```

Expressions

INDEXING AND X-REF (6/6)

```
#include <stdio.h>

int Fibonacci(int);
int Fibonacci(int n) {
  if (n == 0)
    return 0;
  else if (n == 1)
    return 1;
  else
    return (Fibonacci(n-1) + Fibonacci(n-2));
}

int main() {
  int n = 12, i = 0, c;
  for ( c = 1 ; c <= n ; c++ ) {
    printf("%d\n", Fibonacci(i)); i++;
  }
  return 0;
}
```

Operators

CRANGE

FRONTEND

- ~~C++~~
- ~~libclang + Python~~
- libclang + Python + multiprocessing
- crtags: generating tag db
- crange: query

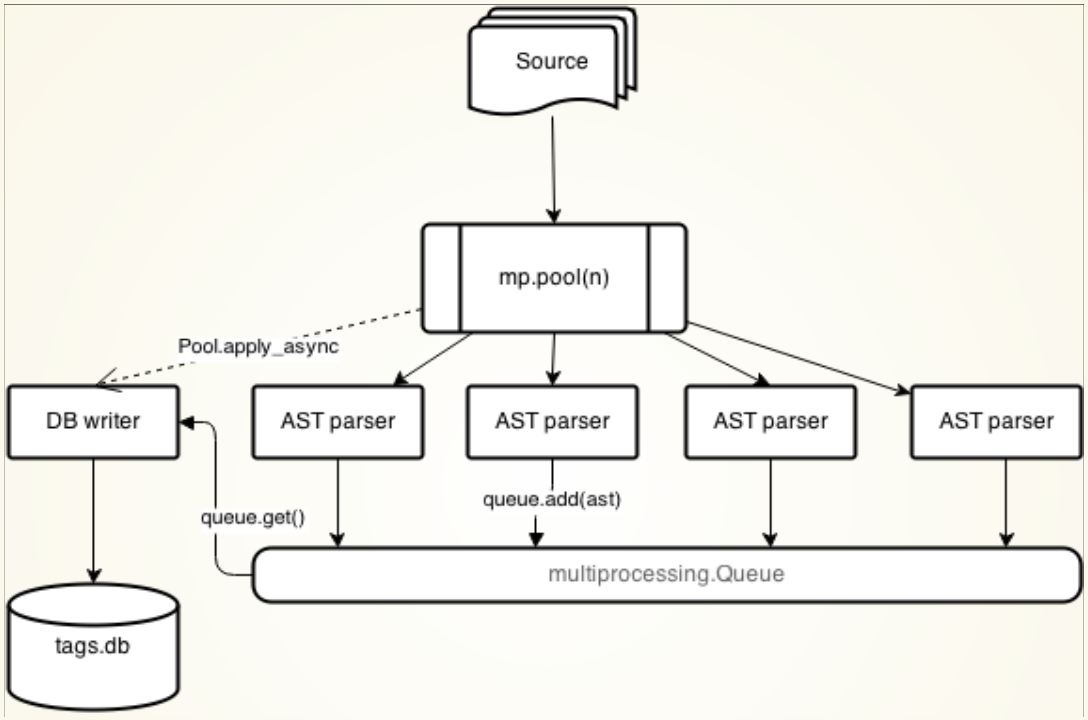
DATASTORE

- ~~JSON~~
- ~~HDF5~~
- ~~BerkeleyDB~~
- SQLite

MULTIPROCESSING MODEL

- 1 DB worker
- 1 multiprocessing.Queue
- n-1 AST workers*

MULTIPROCESSING.POOL()



1 async DB worker, n-1 sync AST workers

CRTAGS

```
$ cd linux-3.13.5
$ time crtags -v -j 32 .
Parsing fs/xfs/xfs_bmap_btree.c (count: 1)
Indexing fs/xfs/xfs_bmap_btree.c (nodes: 379, qsize: 0)
...
Parsing sound/soc/codecs/ak4641.h (count: 34348)
Generating indexes

real    415m10.974s
user    8108m52.241s
sys     72m51.554s
$
```

Generates 22GB tags file in ~7 hours

16GB RAM & 32 CPUs

CURL DEMO

$ time crtags -v -j 32 curl-7.35.0/

CHALLENGES

INCLUDE PATHS

```
$ clang -I/usr/include -Iinclude -I`llvm-config --cflags`
```

MACROS

```
-D_DEBUG -D_GNU_SOURCE -D_CONSTANT_MACROS
```

MAKEFILE/CMAKE

```
[
  { "directory": "/home/anurag/llvm/build",
    "command": "/usr/bin/clang++ -Irelative -DSOMEDEF=\"With spaces, quotes a
    "file": "MachineVerifier.cpp" },
  ...
]
```

PYTHON VS C++

SUMMARY

CRANGE CAN PROVIDE

- Definitions
- Declarations
- References
- Expressions
- Operators
- Symbols
- Source range
- Fancy search
- Code folding*
- Find similar code*

FURTHER READING

- https://github.com/crange/crange
- http://clang.llvm.org/get_started.html
- github.com/trolldbois/python-clang/tree/master/tests/cindex
- https://docs.python.org/2/library/multiprocessing.html
- http://clang.llvm.org/doxygen/group__CINDEX.html

END